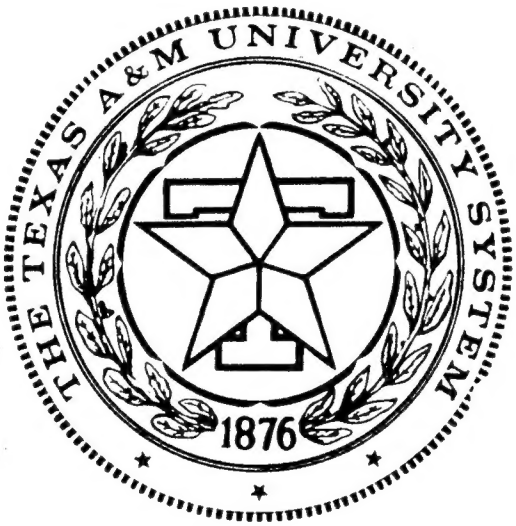
TEXAS A&M UNIVERSITY

Hardware Implementation of a Desktop Supercomputer for High Performance Image Processing

ONR Grant Number N00014-94-1-0516

Technical Report
Nov/01/95-Feb/01/95



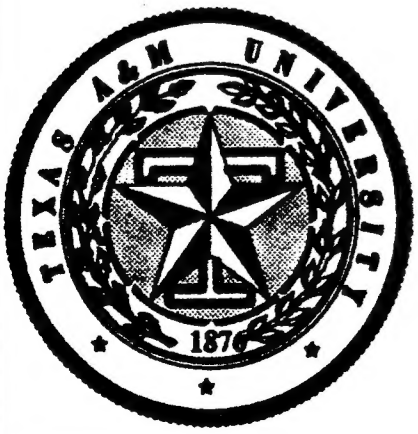
DEPARTMENT OF ELECTRICAL ENGINEERING
College Station, Texas

Hardware Implementation of a Desktop Supercomputer for High Performance Image Processing

ONR Grant Number N00014-94-1-0516

Technical Report
Nov/01/95–Feb/01/95

ROBUST TESTING OF CELLULAR NEURAL NETWORKS



Dr. Jose Pineda de Gyvez

Texas A&M University
Microelectronics Group
Department of Electrical Engineering
College Station, TX, 77843

Phone: (409) 8457477
FAX: (409) 8457161
Email:gyvez@pineda.tamu.edu

Table of Contents

1. Introduction 1

2. Behavioral Aspects 1

3. Electrical Topology 2

4. Classification Of Circuit Faults 4

5. Behavioral Fault Testing 4

6. Fault Identification 7

7. Parametric Fault Testing 8

8. Fault Case Studies 10

9. Conclusions 12

10. References 12

11. Appendix 13

1. Introduction

Cellular Neural Networks (CNN's) are a special class of continuous time neural networks often used for real time image processing. The two dimensional CNN has proved useful for edge detection, noise removal, image thinning, hole filling, image filtering, motion detection, and character recognition[1]. The robust design of CNN's often requires a large number of simulations for design verification. Unfortunately, the circuit representations of the CNN are large and require much CPU time to simulate circuit faults. We propose a CAD tool that generates SPICE macromodels to simulate faults in the hardware of two different CNN topologies. These topologies are the voltage mode and the current mode CNN. The electrical parameters of each macromodeled element are annotated with parameters to match real hardware implementations. The complete CNN macromodel is then simulated using SPICE. The results of the simulation are used to characterize the behavior of a CNN and it's sensitivity to parameter variation, among other things. In other words, the macromodeling fault generator offers the capability for parametric and catastrophic fault simulations. Chua [2] originally introduced a piecewise linear SPICE simulator for simulating the CNN architecture. His model provided the capability for simple CNN simulations, but lacked the ability to model the interconnection impedances of real hardware architectures. Lee [3] and Varientos [4] both presented software CNN behavioral simulators that yield quick results, but that lack the ability to model any of the electrical parameters found in hardware implementations. Macromodels provide the capability to model offsets, impedance mismatches, and other circuit nonidealalities. The reduced complexity of the proposed CNN macromodels allow results to be obtained more quickly than a complete transistor level circuit model. Simulating macromodeled CNN arrays provides an attractive compromise between behavioral simulations and complete circuit simulations.

2. Behavioral Aspects

A two-dimensional CNN array contains MxN locally connected cells. A 5x5 CNN array is shown in Fig. 1. The CNN array is controlled by choosing the initial conditions, setting inputs, and selecting the desired mode of operation. The CNN provides three behavioral modes of operation. They are: i) Cell initialization, ii) Evaluation, and iii) Result extraction. The behavioral modes are selected by two mutually exclusive voltage controlled inputs, *SetIC* and *Eval*, that activate two voltage controlled switches inside the integrator. In the initialization phase, the summer is disconnected from the integrator to allow the state of the cell to start at a known initial condition. The initial condition is selected by activating the signal *SetIC*. When *SetIC* is activated the initial condition voltage *(IC)* is applied directly across the capacitor, which is the physical quantity that represents of the state of the cell. In some applications it is desired to use the input image as the initial condition of the array. For this case, the *IC* signal is connected to the *IN11* signal of each cell. The evaluate mode is entered when the *Eval* signal is activated which connects the summer to the integrator and allowing the dynamic processing to commence. If neither *SetIC* nor *Eval* is activated, the output stage is separated from the integrator and ideally the state of the cell is held indefinitely.

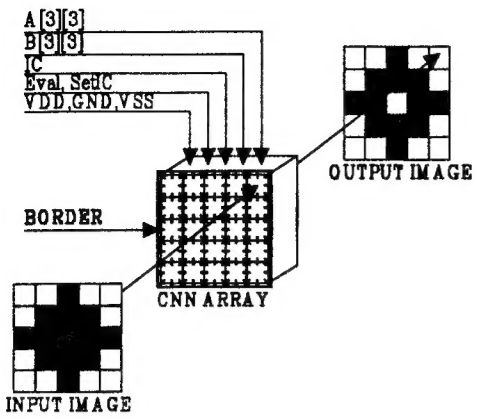


Fig. 1 - CNN Array Block Diagram

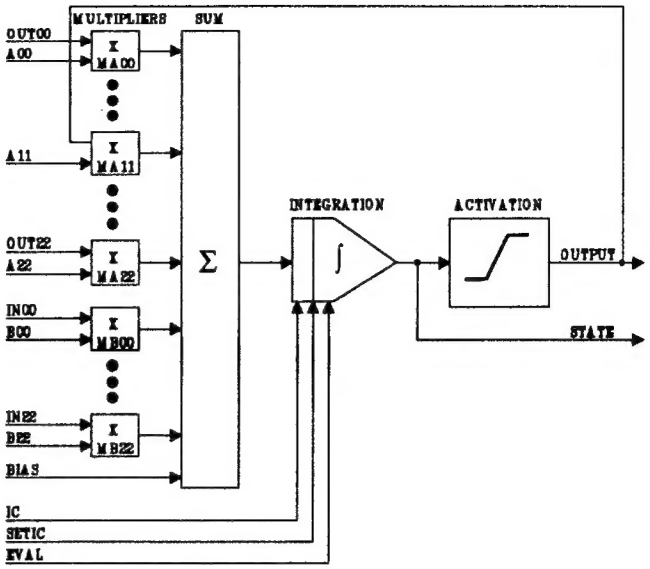


Fig. 2 - CNN Cell Macromodel Block Diagram

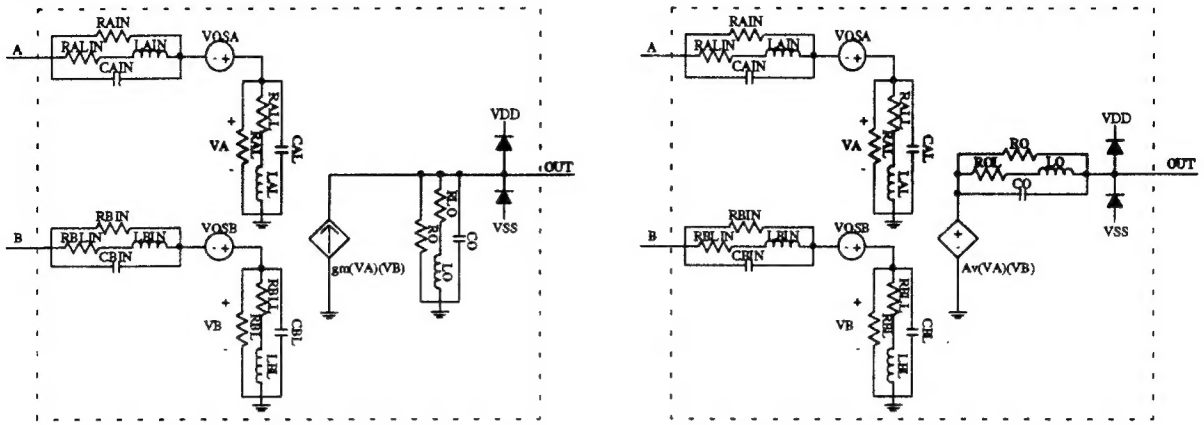
3. Electrical Topology

The macromodeled CNN cell contains 42 voltage inputs and 1 voltage output as shown in Fig. 2. The cell has both global and local voltage inputs. The global inputs are common to all cells and include the feedback template (*A00-A22*), the control template (*B00-B22*), the bias value (*Bias*), the mode control values (*SetIC* and *Eval*), the initial condition value (*IC*), and the power supplies (*VDD*, *GND*, and *VSS*). The local inputs (*IN00-IN22, OUT00-OUT10, OUT12-OUT22*) are unique to each cell and are determined by the location of the cell within the CNN array. Each cell in the array has an activated output voltage (*OUTPUT*) and optionally a pre-activated output voltage (*STATE*). The addition of the STATE output is motivated by Pineda's research into the application of a multilayer CNN used to process color images[5]. The cells located on the outer edges of the array obtain their *INxx* and *OUTxx* inputs from the *Border* value, which is added to the basic CNN architecture to facilitate testing of the array.

The CNN macromodel consists of multipliers, a summer, an integrator, and an activation function. The internal processing of the cell may take place in either the current or voltage domains. Current mode operation is characterized by the use of transconductance multipliers, a current summer, and a current integrator. The voltage mode CNN makes use of voltage multipliers, a voltage summer, and a voltage integrator. In either case, the inputs and outputs of the CNN macromodel are voltages. Each element of the macromodel contains complex input/output impedance's to simulate interconnection parasitics, voltage sources to simulate offsets, and the corresponding elements to perform its original function. The use of the generic complex impedance block is considered a single element in the fault analysis. The complex impedance block may be used to introduce poles and zeros into the transfer function of each macromodeled block.

The current mode (transconductance) and the voltage mode multiplier are shown in Fig. 3. Current mode multipliers are implemented using a voltage controlled current source with an annotated transconductance value, *gm*. Voltage mode multipliers are based upon voltage controlled voltage sources with an annotated voltage gain, *Av*. The multiplier input voltages are developed across a line impedance, an offset voltage source, and across a load impedance. One can think of the line impedance as representing the impedance of the path between the output of a cell and the input to the specific

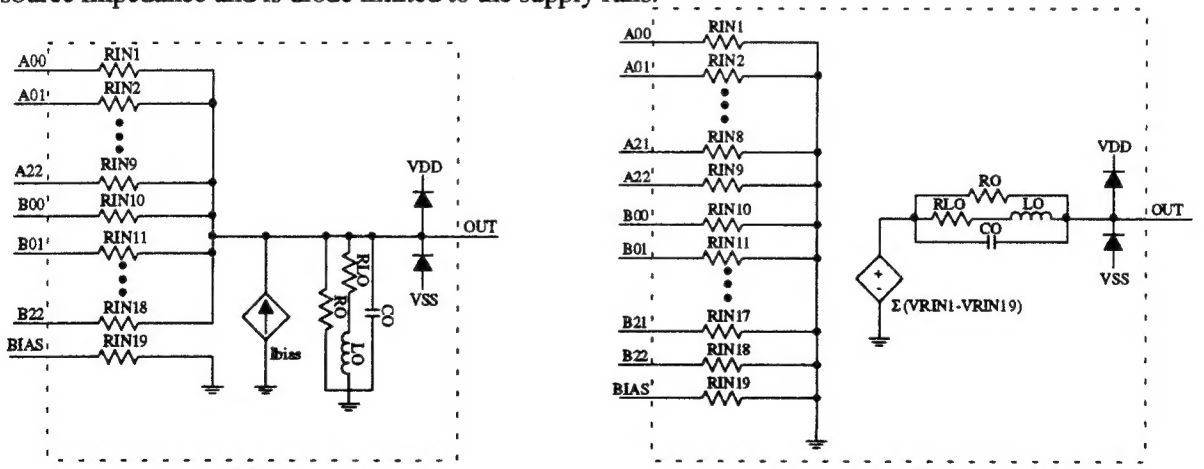
multiplier. The multiplier generates an output that is proportional to the product of the two input load voltages. The current/voltage output is developed across/through a source impedance and then diode limited to the supply range to simulate the saturated region of actual multipliers. Although HSPICE simulator allows dependent sources to be annotated with a minimum and maximum value to model saturation, we used diode limited outputs in our macromodels for SPICE compatibility purposes.



(a) (b)

Fig. 3 - Macromodeled Multipliers - (a) Transconductance Multiplier, (b) Voltage Multiplier

The macromodeled summers shown in Fig. 4 accept the outputs of the 18 multipliers and the *Bias* voltage to produce an output proportional to their sum. This output can be a current or a voltage depending upon the CNN topology. The current mode summer requires that the *Bias* voltage is first converted to a current before the summing with the multiplier currents. The current is summed by a simple wire, developed through a source impedance, and diode limited to avoid unbounded output values. Note that the summer contains a load impedance to provide a DC path to ground when the summer is disconnected from the integrator. The voltage mode summer requires a dependent voltage source that produces the sum of the 19 voltage inputs. The voltage output is developed through a source impedance and is diode limited to the supply rails.



(a) (b)

Fig. 4 - Macromodeled Summers - (a) Current Summer, (b) Voltage Summer

The integrator contains the processing elements of the CNN that primarily determine the dynamics of the array. Fig. 5 shows the block diagram of the current mode and voltage mode integrator. The current mode and the voltage mode integrator both contain two voltage controlled switches. The *SW1* switch is controlled by the *SetIC* signal and when activated forces the state of the cell to the voltage applied at the *IC* input. The *SW2* switch is controlled by the *Eval* signal and when activated connects

the output of the summer to the integrator. If both *SetIC* and *Eval* are inactive, the state of the cell is held until the capacitor discharges. The mode control switches are modeled using ideal voltage controlled resistors that are annotated with "on" and "off" resistance to model real switches.

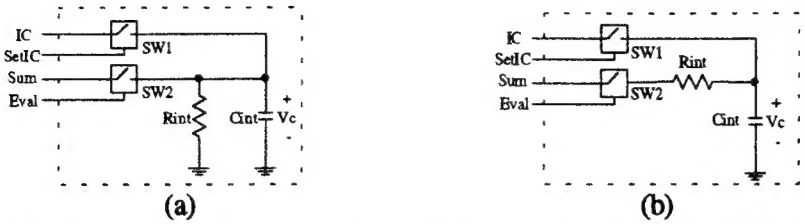


Fig. 5 - Macromodeled Integrators - (a) Current Integrator, (b) Voltage Integrator

The activation function of the CNN is identical in both the current mode and the voltage mode CNN. The activation function is modeled using a polynomial voltage controlled voltage source as shown in Fig. 6. The polynomial source can be adjusted to model any shape activation function, any gain, and any output offset voltage. The output voltage is developed through an output resistance and diode limited to the supply range. Note that the output voltage is internally fed back to the input of feedback multiplier *A11*, eliminating the need for an external *OUT11* input signal. The result is that each CNN cell has only 8 inputs (*OUT00-OUT10* and *OUT12-OUT22*) required from the surrounding cells outputs.

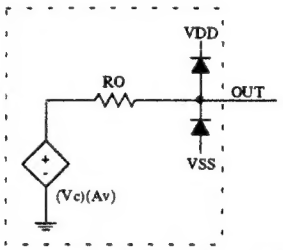


Fig. 6 - Macromodeled Activation Function

4. Classification Of Circuit Faults

We have chosen to classify circuit faults within a CNN array into two categories: hard and soft faults. Hard faults consist of opened, shorted, or stuck-at supply nodes usually resulting from design or processing flaws. Hard faults are easy to detect because the behavior of a faulted array will not match the unfaulted behavior during static tests. Soft faults consist of mismatches in the time constants, offset voltages, and reactive impedance faults usually caused by a miscentered design process. Soft faults are difficult to characterize as they may or may not adversely affect a CNN array depending upon the particular application. A criteria to define a soft fault based upon its adverse effect must be defined. Unfortunately, the adverse effect of a fault depends on the particular CNN application. For this presentation, more than a ± 2% variation in the nominal time constant, more than ± 10% of the power supply in offset voltage, and more than a ± 10% variation in the nominal impedance are considered faults. The behavioral tests are designed to detect hard faults and the parametric tests are designed to detect the soft faults.

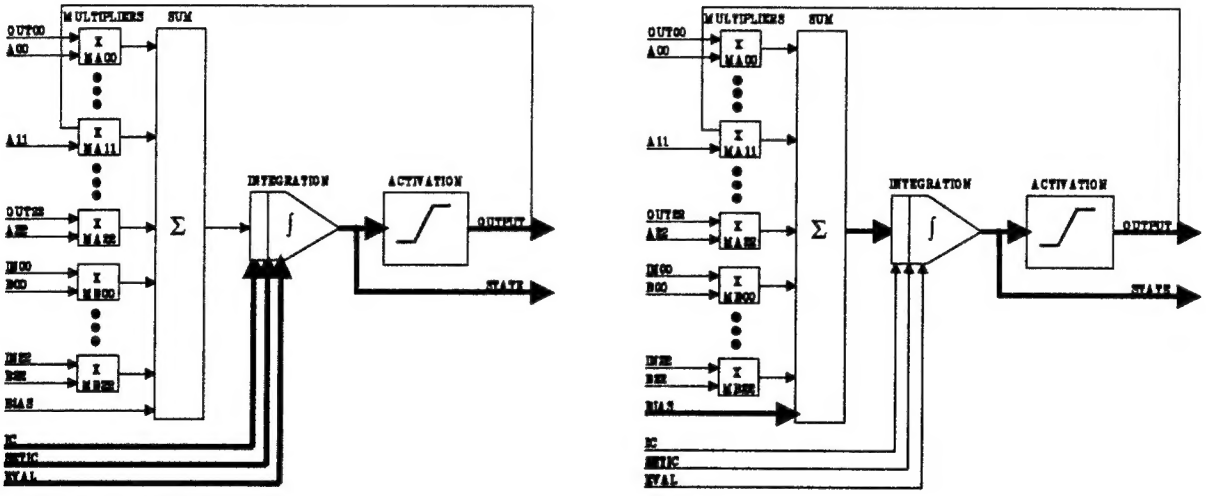
5. Behavioral Fault Testing

A robust testing strategy must provide maximum fault coverage in a minimum number of test vectors. Our macromodeled current and voltage mode CNN cell contains 74 and 94 internal nodes, respectively. Bear in mind that these macromodels were designed to match possible monolithic implementations.

They differ from Chua's basic macromodels[1] in that they try to mimic the actual hardware implementation as opposed to only simulation behavioral effects. Naturally, if the number of cells is multiplied times the number of internal nodes, and the global and local inputs/outputs are added, the magnitude of the testing problem becomes apparent.

Fault detection in a production environment should be comprised of a minimum set of input vectors that produce an output vector set that provides complete visibility of the internal nodes. Visibility in this context refers to the ability to detect if one or more shorted, opened, or stuck-at nodes exist in the array. By comparing a known good output vector set to the output vector set obtained from the untested array, one can determine if the device under test contains any critical faults. From a practical point of view, the behavioral tests must be conducted at intervals much greater than the time constant of the CNN cell to insure proper convergence. For instance, if a circuit had a time constant of 100ns, a valid time interval between test vectors would be selected as 100 times the time constant, or a time interval of 10 μs.

In an effort to simplify the testing methodology, we identified four different testing paths in the CNN architecture. The consequence of this observation is that the behavioral tests are divided into four segments, one for each of the testing paths. These paths are the initial condition path (Fig. 7a), the biasing path (Fig. 7b), the input path (Fig. 7c), and the feedback path (Fig. 7d). The initial condition path and the biasing path of each cell are tested in parallel with no interaction between adjacent cells. The input and feedback path test vectors insure that the multipliers and the interconnections between adjacent cells of the array are fault free. In our procedure the order of the behavioral tests is important as each successive test requires no previous faults. If a static test reveals a fault, all testing should be halted and the path containing the fault must be marked as faulty.



(a) **(b)**

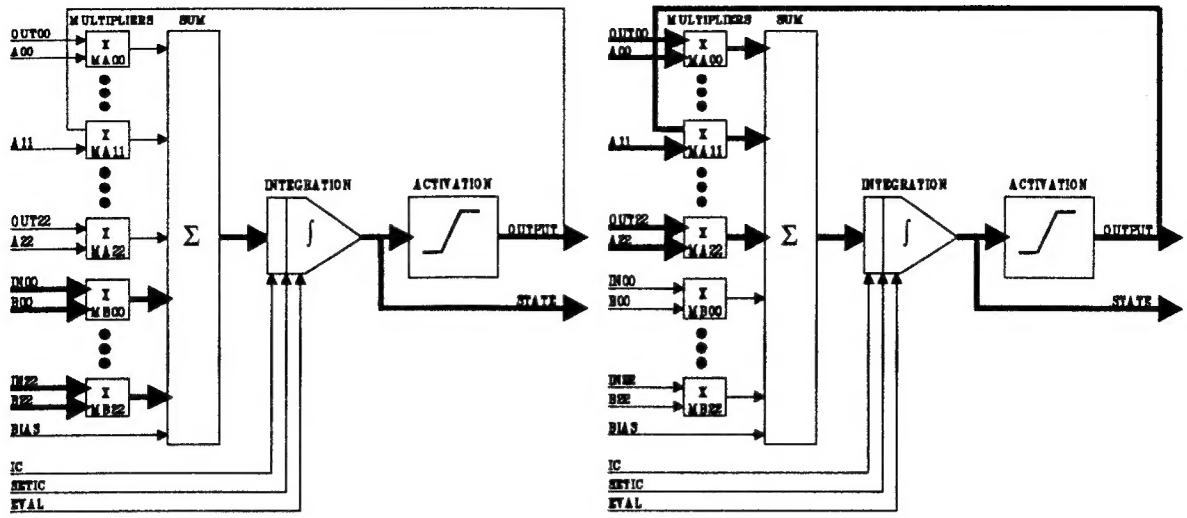


Fig. 7 - Testing Paths - (a) Initial Condition, (b) Biasing, (c) Input, (d) Feedback

The input vectors were designed so that every node in the array is toggled and the result is propagated to the output. By comparing the output of a known good array to the output of a device under test, we can identify paths that contain open, shorted, or stuck-at supply nodes. Although the tests cannot isolate the type or exact location of the fault, they do identify the path that contains the fault(s).

When testing the initial condition and bias paths, the expected output is the same for all cells within the array. To insure this characteristic, every element in the A and B templates is set to zero eliminating any interaction between adjacent cells. The strategy to test the input path is to change the B template and the input image so that every node in the input path is toggled. The expected output depends on the B template, the input image, and the location of the cell within the array. Testing of the feedback path depends upon the ability to set the A template and the *Border* such that every node in the feedback path is toggled. Naturally, the expected output depends on template A, the *Border* value, and the location of the cell within the array.

Eighteen unique testing templates are used to perform behavioral tests for both the multipliers corresponding to the A and B templates in the CNN array. Each of these templates contains only one non-zero element set to either the positive supply (*VDD*) or the negative supply *(VSS)*. The testing templates were designed to propagate the inputs/outputs directionally so that every node is toggled and the multiplier's four quadrant operation is validated. The same templates are used for testing both the A and B multipliers, with different expected results. The template, location, and polarity of the non-zero element is defined by the notation Array:Polarity:Row:Column. All array element numbering is zero based. The top left hand corner element of the array is element [0,0] and the bottom right hand corner element is element [(M-1),(N-1)]. For example, testing template AN21 consists of all elements of the A template set to ZERO except for the third row, second column element that is set to *VSS*. When testing the input path, the expected output image is a propagated version of the input image. Note that if negative multiplier values are used, the expected output image is the inverted and propagated input image. When testing the feedback path, the expected output image is the result of the Border value propagated across the array. When negative multiplier values are used, the *Border* value is inverted as it is propagated across each cell in the array.

6. Fault Identification

The first behavioral test segment (vectors 1-10) verifies the initial condition selection path. All test vectors presented in this section are located in Appendix 1. The purpose of the test is to determine if the initial condition can be properly initialized and extracted. The test requires manipulation of *SetIC* and *IC*. The feedback template, the control template, the *Bias* value, the *Border* value, the *Eval value*, and the *INxx* values are all set to *ZERO* to prevent the cell from having inadvertent initial states. The initial condition is set to the desired value by activating *SetIC* and applying the desired DC voltage to the *IC* pin of the array. The *IC* value charges the state capacitor to the desired value via *SW1*. The array is switched to hold mode by deactivating *SetIC* and the *IC* value is toggled to the two other possible DC initialization values to insure no interaction between the initialized state and the deselected *IC* value. The test verifies that each cell's *SetIC*, *IC*, integrator switch *SW1*, state node, and the activation functional block are all fault free. The process is repeated for each of the possible initial condition values (*VDD*, *GND*, and *VSS*). If any of the initial condition paths contained shorted, opened, or stuck-at faults, the output vector set would differ from the known good output vector set. For example, if a stuck-at *VDD* fault were introduced into a cell's output, test vector #1 would indicate a difference between the actual output (*VDD*) and the expected value of the output (*ZERO*).

The second behavioral test segment (vectors 11-14) insures that the *Bias* signal correctly influences the output of each cell. The A template, the B template, the *Border* value, and the *INxx* values are set to *ZERO*. All cells are initialized to *MINUSV* and the *Bias* signal is changed from *PLUSV*, to *ZERO*, to *MINUSV*. The output of the cell should transition from *VDD*, to *GND*, to *VSS*. The test verifies that the *Bias* input, the bias path through the summer, and the summer to integrator path do not contain any static faults. For example, if a node in the *Biasing* path was opened, test vector #11 would indicate a difference between the actual output (*GND*) and the expected value of the output (*PLUSV*).

The third behavioral test segment (vectors 15-66) insures that the input path and the B multipliers are fault free. Template A, the *Border* value, and the *Bias* values are all set to *ZERO*. The *Eval* signal is constantly activated for all tests in this segment. The B template is chosen to directionally propagate the input image through each possible cell interconnection in the input path. The input images are chosen in inverted pairs so that during the test four quadrant multiplier operation is verified. The input image pairs consist of a positive/negative checkerboard pattern, alternating rows of black and white, and alternating columns of black and white as shown in Fig. 8.

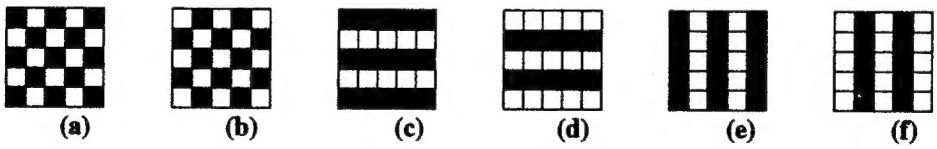


Fig. 8 - Input Testing Images - (a) *IN00* Positive Checker Board, (b) *IN01* Negative Checker Board, (c) *IN02* Positive Rows, (d) *IN03* Negative Rows, (e) *IN04* Positive Columns, (f) *IN05* Negative Columns

Note that in the pseudocode, we define image inversion with a tilde (~). If a tilde precedes an image, all elements are multiplied by -1, inverting the complete image. The resulting output image is the directional propagation of the input image through the selected multiplier path, to the output. The direction of propagation is determined by the non-zero element of the B template as shown in the Appendix. A few examples of the expected output images for this test are shown in Fig. 9 for reference.

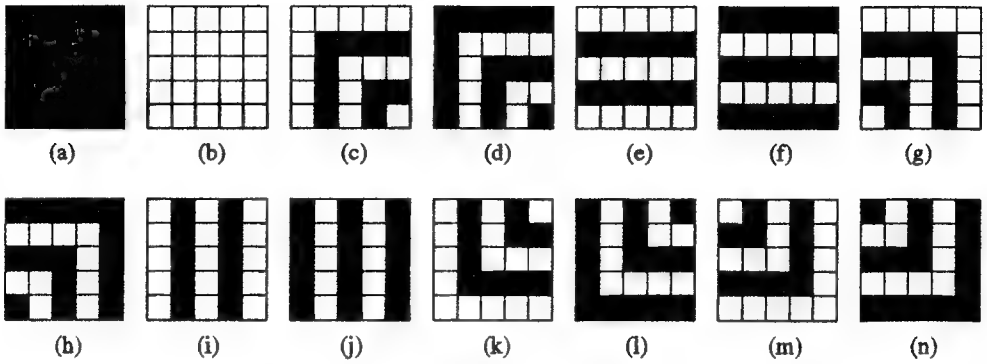


Fig. 9 - Expected Output Images - (a) *OUT00*, (b) *OUT01*, (c) *OUT02*, (d) *OUT03*, (e) *OUT04*, (f) *OUT05*, (g) *OUT06*, (h) *OUT07*, (i) *OUT08*, (j) *OUT09*, (k) *OUT10*, (l) *OUT11*, (m) *OUT12*, (n) *OUT13*,

Note that if a cell is gray, it contains a *ZERO* voltage. The *ZERO* voltage is only be seen at cells that receive their inputs from the *Border* value. The test verifies that the input image path, the control multipliers, and the multiplier to summer paths do not contain any faults. As an example consider the case when the output of the multiplier B00 is shorted to *IC*. Test vector #11 is used to detect the fault by indicating a difference between the actual output (*GND*) and the expected value of the output (*MINUSV*).

The fourth behavioral test segment (vectors 67-96) insures that the feedback path and the A multipliers are fault free. Template B, the *IC* value, and the *Bias* values are all set to *ZERO*. The *Eval* signal is constantly activated for all tests in this segment. The A template is chosen to directionally propagate the *Border* value though each possible cell interconnection in the feedback path. As discussed previously, the *Border* value is inverted to insure proper four quadrant operation of the A multipliers. The resulting output image is the directional propagation of the *Border* through the selected multiplier path, to the outputs. The direction of propagation is determined by the non-zero element of the A template as shown in the Appendix. The test verifies that the feedback path, the feedback multipliers, and the multiplier to summer paths do not contain any faults. If the feedback path contained nodes that are shorted, opened, or stuck-at a DC supply voltage, the output vector set would differ from the known good output vector set. For example, if the output of multiplier A00 in cell (2,2) were stuck-at *GND*, test vector #67 would indicate a difference between the actual output (*GND*) and the expected value of the output (*PLUSV*).

7. Parametric Fault Testing

The parametric tests are designed to reveal excessive time constant mismatches, mismatched impedances, and voltage offsets. The tests require visibility of the non-saturated state of the array. If the cells within the array are equipped with STATE outputs, the tests make use of the full dynamic range of the array. If the cell only has the OUTPUT signal, the dynamic tests extract their information

when the state of the cell is in the linear region of the activation function. The state voltage range that results in an output in the linear range is defined as the *linear input range*. The capability to set the STATE to a known, non-saturated value allows us to determine if there is a fault. The B multipliers (control) are easily tested because we can control both of the inputs which are external to the array. The test sets the inputs of the multiplier to a fixed value and compares the output of the cell to the expected output voltage. The difference between the expected output and the actual output give a measure of the offset and impedance faults contained within each cell. The A multipliers (feedback) present a more difficult testing challenge because we no longer have independent control of each of the multiplier inputs. One of the inputs to the feedback multipliers is always obtained from the output of a cell within the immediate neighborhood. The *Border* is directionally propagated across the array and each cell in the propagation chain is measured to give a measure of the sum of the offsets and impedance faults in the chain. Faults detected using this method can not be easily localized.
The parametric tests are composed of three segments. The first parametric test segment verifies the time constants of each cell in the CNN array. The A template, the B template, and the *Border* value are fixed at *ZERO*. The cell is initialized to *GND* and a step is applied to the *Bias* input from *GND* to *VDD*. The cell output is sampled every 0.1 τ seconds for 10 τ seconds. The result of the test will indicate if each cell's time constant is within 1% of the specifications. The expected output voltage is determined by the equation:

$$V_{OUT} = V_{DD} - (V_{DD})(e^{-t/\tau})$$

For example, if the time constant of all cells in the array is $t_{NOMINAL}$=100ns, except for the center cell with t_{FAULT}=102ns, after 1τ the unfaulted output voltage is Vout=3.161V and the faulted output voltage is Vout=3.197V. The test reveals the time constant fault because the faulted output requires 2% more time to converge to the final value of *VDD*.

The second parametric test segment checks for excessive impedance mismatches, non-linearity, and offsets in the B multipliers. Each multiplier line impedance should be much smaller than the load impedance. If it is not, we can detect a difference in the expected output voltage. If we zero both the A and B template, except for setting one of the B template elements to a fixed positive value, we can apply inputs that result in output voltages that are in the linear region of the activation function. When one of the multipliers inputs is fixed, the output will track the other multiplier input. The input is slowly swept (>100t) in the *linear input range* and multiple data points are sampled from either the OUTPUT or STATE outputs. The test is repeated for the other combination of holding one input constant while sweeping the other input. For example, if the supply voltages were VDD = |VSS| = 5.0 V and the activation slope Av=2 were chosen, the input range that results in the output in the linear region is from -2.5 V to +2.5 V. The input image voltage is fixed at +1.0V and the B template input is swept from -2.5 V to +2.5 V in 50 mV increments resulting in 100 data points. The test is repeated for the B template fixed at +1.0V and the input image swept from -2.5 V to +2.5 V in 50 mV increments. The data is analyzed to insure the output voltage is more than 90% of the input voltage and that the output is within 10% of fault free expected result. Although we can detect offset and impedance faults, we cannot discriminate between them.

The third parametric test segment checks for faults in the A multipliers. If both the A and B template are set to *ZERO*, except for setting one of the eight peripheral elements set to a fixed positive value, we can select a *Border* value that results in an output voltage that is in the linear region of the activation function. The center template value (*A11*) is excluded from this test since the feedback from a cell to itself would result a saturation or oscillation condition occurring after the evaluation phase has commenced. The *Border* is slowly swept (>100t) in the *linear input range* and multiple data points are sampled from either the OUTPUT or STATE output. For example, the A template is fixed at +1.0V

and the *Border* template input is swept from -2.5 V to +2.5 V in 50 mV increments resulting in 100 data points. The data is analyzed to insure the output voltage is more than 90% of the input voltage and that the output is within 10% of fault free expected result.

8. Fault Case Studies

To show the versatility of the CNN testing strategy, we analyze the operation of a 5x5 voltage mode CNN using a hardware macromodel processing an image using an edge detection template. The macromodel was generated using CNNSPICE, a program developed for generating MxN CNN array macromodels. The integration resistor was chosen to be 100K ohms and the integration capacitor was chosen to be 1pF resulting in a time constant of t=100ns. The supplies were selected to be *BLACK*=*VDD*=(+5.0V) and *WHITE*=*VSS*=(-5.0V). An activation slope of *Av*=2 was chosen for this simulation to insure stable outputs when the absolute value of the state of the cell is greater than 2.5 volts. In the voltage mode CNN, the sum of the output resistance of the voltage summer and the integration resistor, along with the integration capacitance, determine the time constant of the array. Typically, the output impedance of the voltage summer is small, thus the integration resistance dominates.

In this presentation, we will introduce four behavioral faults and three parametric faults into our voltage mode macromodel of a CNN. All of the faults were introduced into the center cell of the array for simplicity. The faults consist of: (1) *SetIC* stuck at *VDD*, (2) Integrator input shorted to *IC*, (3) Summer output opened, (4) Feedback multiplier A11 output stuck-at VSS, (5) a 2% time constant fault, (6) a 10% offset voltage in input 1 of multiplier B11, and (7) a excessive impedance mismatch where the line impedance is 10% of the load impedance at the input of input 1 of multiplier A01. The testing results clearly show that behavioral fault detection is possible. Fig. 10 shows the comparison between the unfaulted and the *SetIC* stuck at *VDD* fault case. Based on vectors 2 through 10 the expected output is *ZERO* at time 15μs, *VDD* at time 40μs, and *VSS* at time 70μs. The faulted output clearly indicates a discrepancy at times 15μs-30μs, 40μs-60μs, and 70μs-90μs indicating that a fault exists in the initial condition path. Fig. 11 shows the comparison between the unfaulted case and the integrator input shorted to *IC* fault case. Using vectors 11 through 14 the expected output is *VSS* at time 105μs, *GND* at time 110μs, and *VDD* at time 120μs. The faulted output clearly indicates a discrepancy at times 105μs-110μs and 120μs-135μs indicating that a fault exists in the biasing path. Fig.12 shows the comparison between the unfaulted and the summer output opened fault case. The expected output is *VSS* at time 150μs, *VDD* at time 160μs, and *VSS* at time 180μs according to vectors 15 through 22. The faulted output clearly indicates a discrepancy at times 145μs-215μs indicating a fault exists in the input image path. Fig. 13 shows the comparison between the unfaulted and the feedback multiplier A11 output stuck-at *VSS* fault case. The expected output is *VDD* at time 790μs, *ZERO* at time 800μs, and *VDD* at time 810μs. The faulted output clearly indicates a discrepancy at time 785μs-800μs and 810μs-815μs indicating a fault exists in the feedback path.

The dynamic test results revealed faults that exist in the feedback path. Fig. 14 shows the comparison between the unfaulted and a 2% time constant fault. The expected output after 1.3t is Vout=3.637 V. The faulted output has only reached Vout=3.6V after 1.3t indicating a 2% time constant mismatch. Fig. 15 shows the comparison between the unfaulted and a 10% offset fault in A input of the B11 multiplier. When sweeping the A input of multiplier in the *linear input range*, the output clearly indicates a difference between the expected values between -2.5V to +2.5V and the actual output voltages of -1.25V to +1.25V. Fig. 16 shows the comparison between the unfaulted and a impedance fault in A input of the A01 multiplier. The faulted output clearly indicates a 10% difference between the expected and actual output voltages.

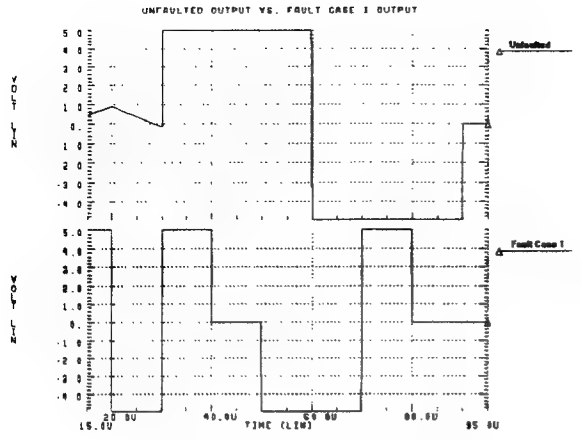


Fig. 10 - Static Fault Case 1

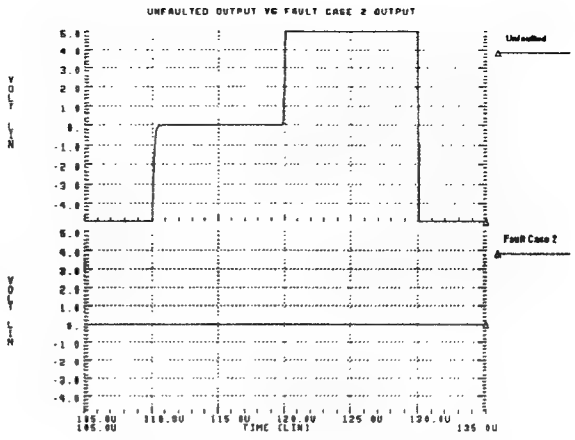


Fig. 11 - Static Fault Case 2

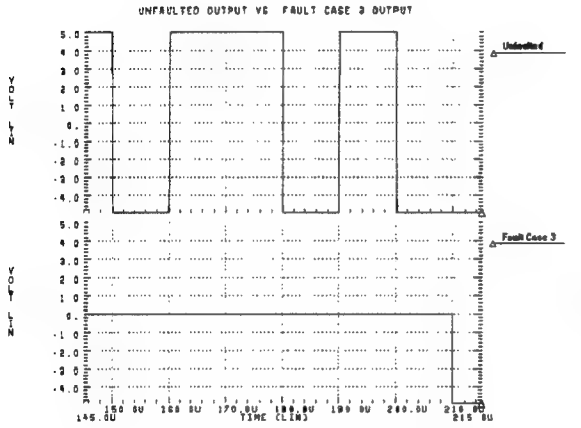


Fig. 12 - Static Fault Case 3

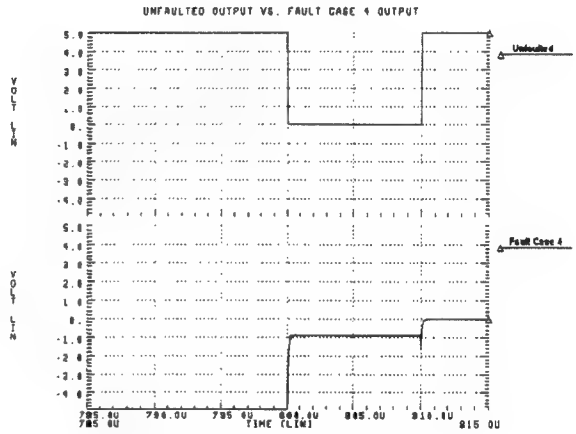


Fig. 13 - Static Fault Case 4

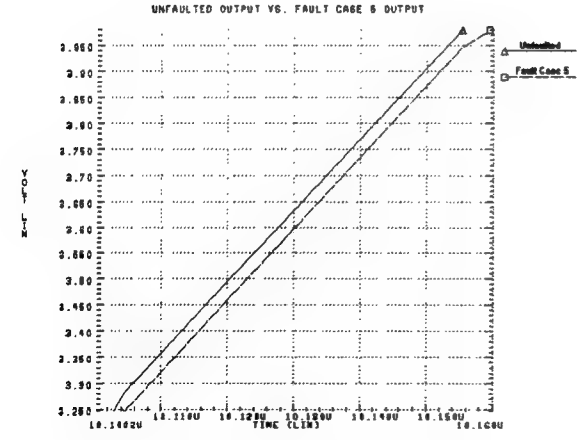


Fig. 14 - Dynamic Faulty Case 1

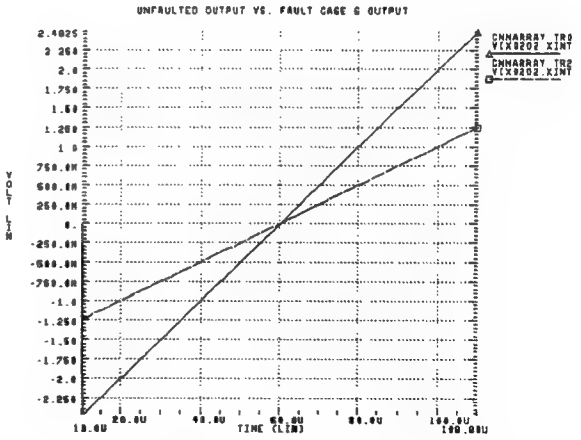


Fig. 15 - Dynamic Fault Case 2

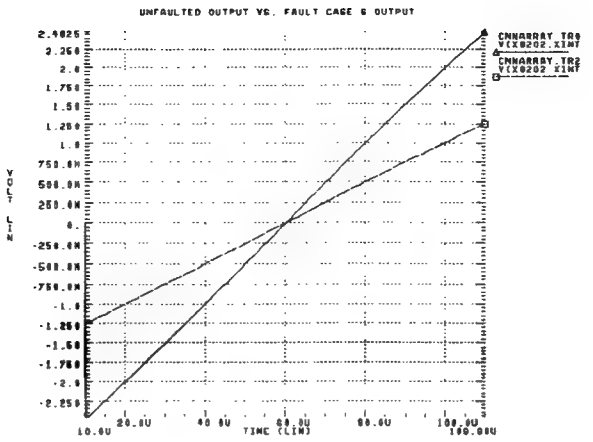


Fig. 16 - Dynamic Fault Case 3

9. Conclusions

In this report we have presented a strategy for macromodeling hardware implementations of two-dimensional CNN's. We have shown that the macromodels can be useful when developing testing strategies for VLSI implementations of CNN arrays by presenting a strategy for testing hardware implementations of two-dimensional CNN's. We have demonstrated that the tests provide extensive fault coverage independent of the circuit topology. We improved the testability of the CNN by breaking the cell into a four different testing paths. Detailed test vectors were provided to demonstrate the effectiveness of the testing strategy. The parametric tests were developed based upon the properties inherent to the CNN architecture. The testing strategy was used to analyze the effect of seven different faults introduced into a macromodel of a 5x5 voltage mode CNN array. The macromodel simulation results prove that faults may be detected by applying the test methods presented in this report.

10. References

[1] L.O. Chua and L. Yang, "Cellular Neural Networks: Applications," *IEEE Trans. Circuits Syst.*, vol. 35, pp.1273-1290, 1988.

[2] L.O. Chua and L. Yang, "Cellular Neural Networks: Theory," *IEEE Transactions on Circuits and Systems*, vol. 35, pp.1257-1272, 1988.

[3] C. C. Lee and J. Pineda de Gyvez, "Single-Layer CNN Simulator," *IEEE International Symposium on Circuits and Systems*, pp.217-220, London, 1994.

[4] J.E. Varrientos and E. Sanchez-Sinencio, "CELLSIM: A CNN Simulator for the Personal Computer," *Proc. of 35th Midwest Symposium on Circuits and Systems*, pp.1384-1387, Washington, 1988.

[5] J. Pineda, "XCNN: A Software Package For Color Image Processing," *Proc. IEEE Intl. Workshop Cellular Neural Networks Appl., CNNA-94*, Rome, 1994.

11. Appendix

Static Test Segment # 1 Vectors - Test Initial Condition Path

Vector	Time	A[00:22]	B[00:22]	Bias	SetIc	Eval	Border	Input	IC	Expected Output
1	0u	ZERO,	ZERO,	ZERO,	LOW,	LOW,	ZERO,	ZERO,	ZERO	ZERO
2	10u	ZERO,	ZERO,	ZERO,	HIGH,	LOW,	ZERO,	ZERO,	ZERO	ZERO
3	20u	ZERO,	ZERO,	ZERO,	LOW,	LOW,	ZERO,	ZERO,	PLUSV	ZERO
4	30u	ZERO,	ZERO,	ZERO,	LOW,	LOW,	ZERO,	ZERO,	MINUSV	ZERO
5	40u	ZERO,	ZERO,	ZERO,	HIGH,	LOW,	ZERO,	ZERO,	PLUSV	PLUSV
6	50u	ZERO,	ZERO,	ZERO,	LOW,	LOW,	ZERO,	ZERO,	ZERO	PLUSV
7	60u	ZERO,	ZERO,	ZERO,	LOW,	LOW,	ZERO,	ZERO,	MINUSV	PLUSV
8	70u	ZERO,	ZERO,	ZERO,	HIGH,	LOW,	ZERO,	ZERO,	MINUSV	MINUSV
9	80u	ZERO,	ZERO,	ZERO,	LOW,	LOW,	ZERO,	ZERO,	PLUSV	MINUSV
10	90u	ZERO,	ZERO,	ZERO,	LOW,	LOW,	ZERO,	ZERO,	ZERO	MINUSV

Static Test Segment # 2 Vectors - Test Bias Path

Vector	Time	A[00:22]	B[00:22]	Bias	SetIc	Eval	Border	Input	IC	Expected Output
11	100u	ZERO,	ZERO,	ZERO,	HIGH,	LOW,	ZERO,	ZERO,	MINUSV	MINUSV
12	110u	ZERO,	ZERO,	PLUSV,	LOW,	HIGH,	ZERO,	ZERO,	ZERO	PLUSV
13	120u	ZERO,	ZERO,	ZERO,	LOW,	HIGH,	ZERO,	ZERO,	ZERO	ZERO
14	130u	ZERO,	ZERO,	MINUSV,	LOW,	HIGH,	ZERO,	ZERO,	ZERO	MINUSV

Static Test Segment # 3 Vectors - Test Input and Control Paths

Vector	Time	A[00:22]	B[00:22]	Bias	SetIc	Eval	Border	Input	IC	Expected Output
15	140u	ZERO,	BP00,	ZERO,	LOW,	HIGH,	ZERO,	IN02,	ZERO	DownRight(IN02)
16	150u	ZERO,	BP00,	ZERO,	LOW,	HIGH,	ZERO,	IN03,	ZERO	DownRight(IN03)
17	160u	ZERO,	BP00,	ZERO,	LOW,	HIGH,	ZERO,	IN04,	ZERO	DownRight(IN04)
18	170u	ZERO,	BP00,	ZERO,	LOW,	HIGH,	ZERO,	IN05,	ZERO	DownRight(IN05)
19	180u	ZERO,	BN00,	ZERO,	LOW,	HIGH,	ZERO,	IN02,	ZERO	~DownRight(IN02)
20	190u	ZERO,	BN00,	ZERO,	LOW,	HIGH,	ZERO,	IN03,	ZERO	~DownRight(IN03)
21	200u	ZERO,	BN00,	ZERO,	LOW,	HIGH,	ZERO,	IN04,	ZERO	~DownRight(IN04)
22	210u	ZERO,	BN00,	ZERO,	LOW,	HIGH,	ZERO,	IN05,	ZERO	~DownRight(IN05)
23	220u	ZERO,	BP01,	ZERO,	LOW,	HIGH,	ZERO,	IN02,	ZERO	Down(IN02)
24	230u	ZERO,	BP01,	ZERO,	LOW,	HIGH,	ZERO,	IN03,	ZERO	Down(IN03)
25	240u	ZERO,	BN01,	ZERO,	LOW,	HIGH,	ZERO,	IN02,	ZERO	~Down(IN02)
26	250u	ZERO,	BN01,	ZERO,	LOW,	HIGH,	ZERO,	IN03,	ZERO	~Down(IN03)
27	260u	ZERO,	BP02,	ZERO,	LOW,	HIGH,	ZERO,	IN02,	ZERO	DownLeft(IN02)
28	270u	ZERO,	BP02,	ZERO,	LOW,	HIGH,	ZERO,	IN03,	ZERO	DownLeft(IN03)
29	280u	ZERO,	BP02,	ZERO,	LOW,	HIGH,	ZERO,	IN04,	ZERO	DownLeft(IN04)
30	290u	ZERO,	BP02,	ZERO,	LOW,	HIGH,	ZERO,	IN05,	ZERO	DownLeft(IN05)
31	300u	ZERO,	BN02,	ZERO,	LOW,	HIGH,	ZERO,	IN02,	ZERO	~DownLeft(IN02)
32	310u	ZERO,	BN02,	ZERO,	LOW,	HIGH,	ZERO,	IN03,	ZERO	~DownLeft(IN03)
33	320u	ZERO,	BN02,	ZERO,	LOW,	HIGH,	ZERO,	IN04,	ZERO	~DownLeft(IN04)
34	330u	ZERO,	BN02,	ZERO,	LOW,	HIGH,	ZERO,	IN05,	ZERO	~DownLeft(IN05)
35	340u	ZERO,	BP10,	ZERO,	LOW,	HIGH,	ZERO,	IN04,	ZERO	Right(IN04)
36	350u	ZERO,	BP10,	ZERO,	LOW,	HIGH,	ZERO,	IN05,	ZERO	Right(IN05)
37	360u	ZERO,	BN10,	ZERO,	LOW,	HIGH,	ZERO,	IN04,	ZERO	~Right(IN04)
38	370u	ZERO,	BN10,	ZERO,	LOW,	HIGH,	ZERO,	IN05,	ZERO	~Right(IN05)
39	380u	ZERO,	BP11,	ZERO,	LOW,	HIGH,	ZERO,	IN00,	ZERO	IN00
40	390u	ZERO,	BP11,	ZERO,	LOW,	HIGH,	ZERO,	IN01,	ZERO	IN01
41	400u	ZERO,	BN11,	ZERO,	LOW,	HIGH,	ZERO,	IN00,	ZERO	~IN00
42	410u	ZERO,	BN11,	ZERO,	LOW,	HIGH,	ZERO,	IN01,	ZERO	~IN01
43	420u	ZERO,	BP12,	ZERO,	LOW,	HIGH,	ZERO,	IN04,	ZERO	Left (IN04)
44	430u	ZERO,	BP12,	ZERO,	LOW,	HIGH,	ZERO,	IN05,	ZERO	Left (IN05)
45	440u	ZERO,	BN12,	ZERO,	LOW,	HIGH,	ZERO,	IN04,	ZERO	~Left (IN04)
46	450u	ZERO,	BN12,	ZERO,	LOW,	HIGH,	ZERO,	IN05,	ZERO	~Left (IN05)
47	460u	ZERO,	BP20,	ZERO,	LOW,	HIGH,	ZERO,	IN02,	ZERO	UpRight(IN02)
48	470u	ZERO,	BP20,	ZERO,	LOW,	HIGH,	ZERO,	IN03,	ZERO	UpRight(IN03)
49	480u	ZERO,	BP20,	ZERO,	LOW,	HIGH,	ZERO,	IN04,	ZERO	UpRight(IN04)
50	490u	ZERO,	BP20,	ZERO,	LOW,	HIGH,	ZERO,	IN05,	ZERO	UpRight(IN05)
51	500u	ZERO,	BN20,	ZERO,	LOW,	HIGH,	ZERO,	IN02,	ZERO	~UpRight(IN02)
52	510u	ZERO,	BN20,	ZERO,	LOW,	HIGH,	ZERO,	IN03,	ZERO	~UpRight(IN03)
53	520u	ZERO,	BN20,	ZERO,	LOW,	HIGH,	ZERO,	IN04,	ZERO	~UpRight(IN04)
54	530u	ZERO,	BN20,	ZERO,	LOW,	HIGH,	ZERO,	IN05,	ZERO	~UpRight(IN05)

55	540u	ZERO,	BP21,	ZERO,	LOW,	HIGH,	ZERO,	IN02,	ZERO	Up(IN02)
56	550u	ZERO,	BP21,	ZERO,	LOW,	HIGH,	ZERO,	IN03,	ZERO	Up(IN03)
57	560u	ZERO,	BN21,	ZERO,	LOW,	HIGH,	ZERO,	IN02,	ZERO	~Up(IN02)
58	570u	ZERO,	BN21,	ZERO,	LOW,	HIGH,	ZERO,	IN03,	ZERO	~Up(IN03)
59	580u	ZERO,	BP22,	ZERO,	LOW,	HIGH,	ZERO,	IN02,	ZERO	UpLeft(IN02)
60	590u	ZERO,	BP22,	ZERO,	LOW,	HIGH,	ZERO,	IN03,	ZERO	UpLeft(IN03)
61	600u	ZERO,	BP22,	ZERO,	LOW,	HIGH,	ZERO,	IN04,	ZERO	UpLeft(IN04)
62	610u	ZERO,	BP22,	ZERO,	LOW,	HIGH,	ZERO,	IN05,	ZERO	UpLeft(IN05)
63	620u	ZERO,	BN22,	ZERO,	LOW,	HIGH,	ZERO,	IN02,	ZERO	~UpLeft(IN02)
64	630u	ZERO,	BN22,	ZERO,	LOW,	HIGH,	ZERO,	IN03,	ZERO	~UpLeft(IN03)
65	640u	ZERO,	BN22,	ZERO,	LOW,	HIGH,	ZERO,	IN04,	ZERO	~UpLeft(IN04)
66	650u	ZERO,	BN22,	ZERO,	LOW,	HIGH,	ZERO,	IN05,	ZERO	~UpLeft(IN05)

Static Test Segment # 4 Vectors - Test Output and Feedback Paths

Vector	Time	A[00:22]	B[00:22]	Bias	SetIc	Eval	Border	Input	IC	Expected Output
67	660u	AP00,	ZERO,	ZERO,	LOW,	HIGH,	PLUSV,	ZERO,	ZERO	Out00
68	670u	AP00,	ZERO,	ZERO,	LOW,	HIGH,	MINUSV,	ZERO,	ZERO	Out01
69	680u	AN00,	ZERO,	ZERO,	LOW,	HIGH,	PLUSV,	ZERO,	ZERO	Out02
70	690u	AN00,	ZERO,	ZERO,	LOW,	HIGH,	MINUSV,	ZERO,	ZERO	Out03
71	700u	AP01,	ZERO,	ZERO,	LOW,	HIGH,	PLUSV,	ZERO,	ZERO	Out00
72	710u	AP01,	ZERO,	ZERO,	LOW,	HIGH,	MINUSV,	ZERO,	ZERO	Out01
73	720u	AN01,	ZERO,	ZERO,	LOW,	HIGH,	PLUSV,	ZERO,	ZERO	Out04
74	730u	AN01,	ZERO,	ZERO,	LOW,	HIGH,	MINUSV,	ZERO,	ZERO	Out05
75	740u	AP02,	ZERO,	ZERO,	LOW,	HIGH,	PLUSV,	ZERO,	ZERO	Out00
76	750u	AP02,	ZERO,	ZERO,	LOW,	HIGH,	MINUSV,	ZERO,	ZERO	Out01
77	760u	AN02,	ZERO,	ZERO,	LOW,	HIGH,	PLUSV,	ZERO,	ZERO	Out06
78	770u	AN02,	ZERO,	ZERO,	LOW,	HIGH,	MINUSV,	ZERO,	ZERO	Out07
79	780u	AP10,	ZERO,	ZERO,	LOW,	HIGH,	PLUSV,	ZERO,	ZERO	Out00
80	790u	AP10,	ZERO,	ZERO,	LOW,	HIGH,	MINUSV,	ZERO,	ZERO	Out01
81	800u	AN10,	ZERO,	ZERO,	LOW,	HIGH,	PLUSV,	ZERO,	ZERO	Out08
82	810u	AN10,	ZERO,	ZERO,	LOW,	HIGH,	MINUSV,	ZERO,	ZERO	Out09
83	820u	AP11,	ZERO,	ZERO,	HIGH,	LOW,	ZERO,	ZERO,	ZERO	ZERO
84	830u	AP11,	ZERO,	ZERO,	LOW,	HIGH,	ZERO,	ZERO,	ZERO	ZERO
85	840u	AP12,	ZERO,	ZERO,	LOW,	HIGH,	PLUSV,	ZERO,	ZERO	Out00
86	850u	AP12,	ZERO,	ZERO,	LOW,	HIGH,	MINUSV,	ZERO,	ZERO	Out01
87	860u	AN12,	ZERO,	ZERO,	LOW,	HIGH,	PLUSV,	ZERO,	ZERO	Out08
88	870u	AN12,	ZERO,	ZERO,	LOW,	HIGH,	MINUSV,	ZERO,	ZERO	Out09
89	880u	AP20,	ZERO,	ZERO,	LOW,	HIGH,	PLUSV,	ZERO,	ZERO	Out00
90	890u	AP20,	ZERO,	ZERO,	LOW,	HIGH,	MINUSV,	ZERO,	ZERO	Out01
91	900u	AN20,	ZERO,	ZERO,	LOW,	HIGH,	PLUSV,	ZERO,	ZERO	Out10
92	910u	AN20,	ZERO,	ZERO,	LOW,	HIGH,	MINUSV,	ZERO,	ZERO	Out11
93	920u	AP21,	ZERO,	ZERO,	LOW,	HIGH,	PLUSV,	ZERO,	ZERO	Out00
94	930u	AP21,	ZERO,	ZERO,	LOW,	HIGH,	MINUSV,	ZERO,	ZERO	Out01
95	940u	AN21,	ZERO,	ZERO,	LOW,	HIGH,	PLUSV,	ZERO,	ZERO	Out04
96	950u	AN21,	ZERO,	ZERO,	LOW,	HIGH,	MINUSV,	ZERO,	ZERO	Out05
97	960u	AP22,	ZERO,	ZERO,	LOW,	HIGH,	PLUSV,	ZERO,	ZERO	Out00
98	970u	AP22,	ZERO,	ZERO,	LOW,	HIGH,	MINUSV,	ZERO,	ZERO	Out01
99	980u	AN22,	ZERO,	ZERO,	LOW,	HIGH,	PLUSV,	ZERO,	ZERO	Out12
100	990u	AN22,	ZERO,	ZERO,	LOW,	HIGH,	MINUSV,	ZERO,	ZERO	Out13

A Template (Feedback) Propagation Mapping

Non-Zero Directon of Element	Propagation	Expected Output Image
A00	Down/Right	Border propagated down and right
A01	Down	Border propagated down
A02	Down/Left	Border propagated down and left
A10	Right	Border propagated right
A11	None	Hold the last state
A12	Left	Border propagated left
A20	Up/Right	Border propagated up and right
A21	Up	Border propagated up
A22	Up/Left	Border propagated up and left

B Template (Control) Propagation Mapping

Non-Zero Directon of

Element	Propagation	Expected Output Image
B00	Down/Right	Input imaged shifted down 1unit and right 1unit
B01	Down	Input imaged shifted down 1unit
B02	Down/Left	Input imaged shifted down 1 unit and left 1 unit
B10	Right	Input image shifted right 1 unit
B11	None	Input image not shifted
B12	Left	Input image shifted left 1 unit
B20	Up/Right	Input image shifted up 1 unit and right 1 unit
B21	Up	Input image shifted up 1 unit
B22	Up/Left	Input image shifted up 1 unit and left 1 unit

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. AGENCY USE ONLY (Leave blank)	2. REPORT DATE	3. REPORT TYPE AND DATES COVERED
	February 10, 1995	Technical Report 11/10/95 - 2/01/95

4. TITLE AND SUBTITLE
Robust Testing of Cellular Neural Networks

5. FUNDING NUMBERS
G
N99914-94-1-0516

6. AUTHOR(S)
Jose Pineda de Gyvez

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
Texas Engineering Experiment Station
Texas A&M University

8. PERFORMING ORGANIZATION REPORT NUMBER
93-549/#4

9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES)
Office of Naval Research
Code 214: JWK
Ballston Tower One
800 North Quincy Street
Arlington, Virginia 22217-5660

10. SPONSORING / MONITORING AGENCY REPORT NUMBER

11. SUPPLEMENTARY NOTES

12a. DISTRIBUTION / AVAILABILITY STATEMENT
A: Approved for public release: distribution unlimited

12b. DISTRIBUTION CODE

13. ABSTRACT *(Maximum 200 words)*

A method for detecting circuit faults within two dimensional Cellular Neural Network (CNN) arrays is presented. The need to develop robust methods for detecting faults is driven by the lack of visibility of the internal nodes in VLSI implementations of CNN arrays. The method is composed of both behavioral and parametric tests and detects faults independent of the size or topology of the CNN array. The behavioral tests reveal nodes that exhibit opened, shorted, or stuck-at a supply voltage faults. The parametric tests reveal excessive time constant mismatches, impedance mismatches and voltage offsets. Seven fault cases are introduced into a macromodel of a voltage mode CNN array to provide insight to the usefulness of the proposed testing methodology.

14. SUBJECT TERMS
Cellular Neural Networks, Testing

15. NUMBER OF PAGES
16

16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
Unclassified	Unclassified	Unclassified	